Introduction into Information Technology

Lecture contents

- Overview of the course goals
- Lecture schedule, lecturer
- Literature, additional material, references
- Lectureplan
- Different schools of thought
- Basic Concepts

course goals

- Give a compact overview of informatics as a whole.
- Give a historical overview of IT theory, technology and business development
- Introduce future courses for further studies

NB! the course DOES NOT TEACH elementary computer usage skills

what the course gives

Enables to understand more or less during the first semester:

- what is informatics, what is our field of study
- what will be taught in further IT courses
- how different IT topics are related to each other
- what topic is essential for what: in practice and in theory
- how IT has developed up till now and what we can expect in the near future
- what are the hot topics in IT technology and in business

course time, location, evaluation, practice

- ITK:
  - Tuesdays kell 17.45-19.15 IT Kolledzhi loengusaalis
  - Exercises: E.Domiczi
- Lecturer(s) E.Domiczi
- Altogether 15 (?) lectures
- Practice separately
- course concluded with
  - Kolledzh: evaluation
  - To pass one has to
    - complete practical works (not so many)
    - pass successfully the written final exam/evaluation-work
- **http://deepthought.ttu.ee/tunniplaan/2975.html**

T	17.45 - 19.15	17.45 - 19.15 ITX0025 | Introduction to Information Technology | Loeng E. Domiczi | IT-140 + praktikum

course materials and tasks

- the course **DOESN'T HAVE** a single course-book.
- the biggest part of the material appears on the **net** before the lecture or afterwards (usually on the same day).
  There is always available lecturing plan and basic topics, but many explanations and details are missing from the net-version.

  http://www.lambda.ee/index.php/Introduction_to_Information_Technology

- the course material is in English (Estonian language material is available for lectures of previous years).
- in addition to the lectures on the net there are also other materials available on the net, but mostly in English.
- Explanatory details can be obtained only by attending the lectures. Self-study is theoretically possible, but it's time consuming and difficult..
- as practical work of the course (on computer) we mainly use David Eck's exercises and software (to be ascertained).
  http://math.hws.edu/eck/index.html
- other obligatory course tasks are in writing.

Literature applicable to the beginning

- **Literature:**
  - New Hacker's Dictionary.
  - The Cathedral and the Bazaar
  - Gnu manifests
- **Applicable links to be read daily:**
  - www.news.com
  - www.slashdot.org

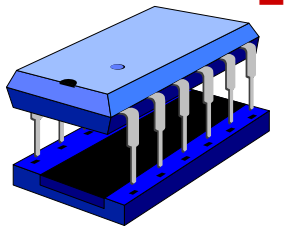
preliminary lecture plan of the course

1. Introduction. Operating principles of computers
2. Early history. Industrialization. Theoretical basics. Logic.
3. Middle history: from WWII till the 80's. programming.
4. Recent history: end of 70's, beginning of 80's. Personal Computers.
5. From the beginning of the 80's till now. Applications, commercial- and free software.
6. Computer structure. programming languages.
7. Operating systems.
8. Software architecture and paradigms.
9. Network software paradigm. Internet applications and technology.
10. Functional and logical programming.
11. Algorithms. Complexity.
12. Algorithms. Solvability.
13. Artificial intelligence.
14. IT management, project lifecycle. Project management.
15. Repetition.

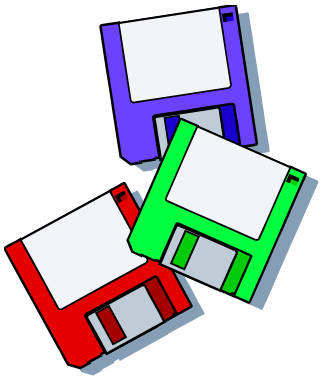
Schools of thought

Hardware ja software

- **Hardware**
  - Physical, tangible parts.
  - Examples: keyboard, display, integrated circuit, etc.



- **software**
  - Programs and data
  - Program is a sequence of instructions



Algorithm and program

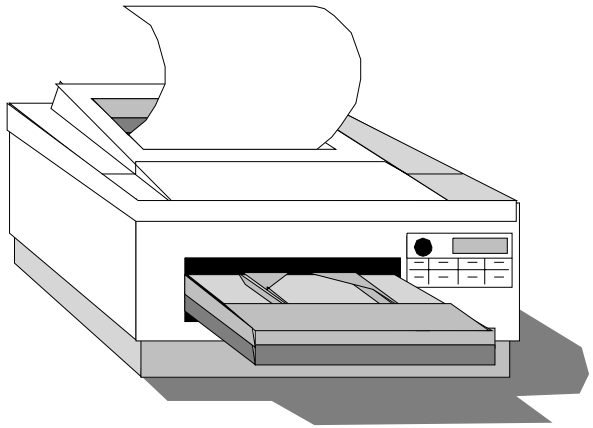
- **Algorithm** is an exact, step-by-step, but not necessarily formal guide to do something. Examples:
  - Food recipe.
  - Guide to solve square-root.
- **Algorithmic problem -** problem, for which the solution can be written as a list of tasks to complete.
- **program** is an algorithm written in a formal, unambiguous language. Computers can execute only programs.

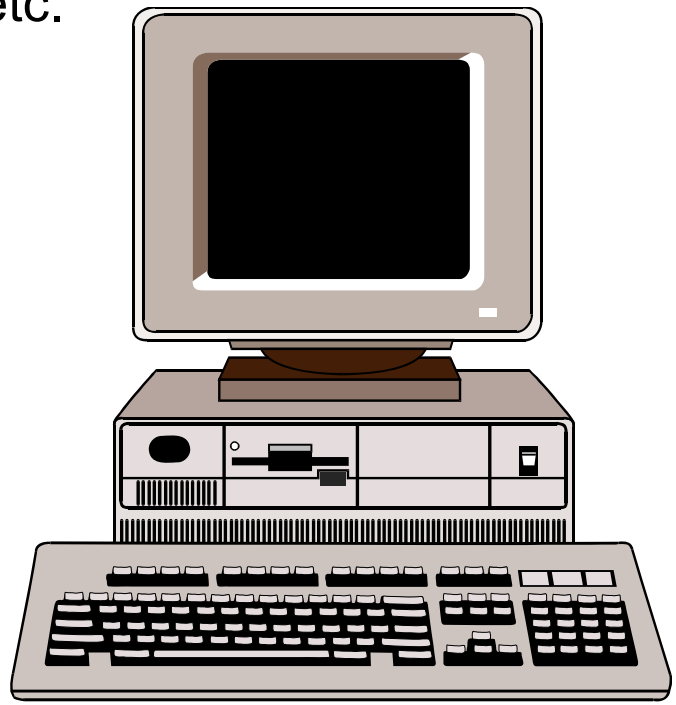
Analogue- and digital system

- **Analogue system**
  - saving (reflecting) of data is proportional
  - For example: thermometer, vinyl record, foto
- **Digital system**
  - (continuous) data chopped-up into pieces, which are separately stored
  - Example: CD, computer program, writing as letters and as bits
- **From one to the other: digitalize, digitize, digitise**

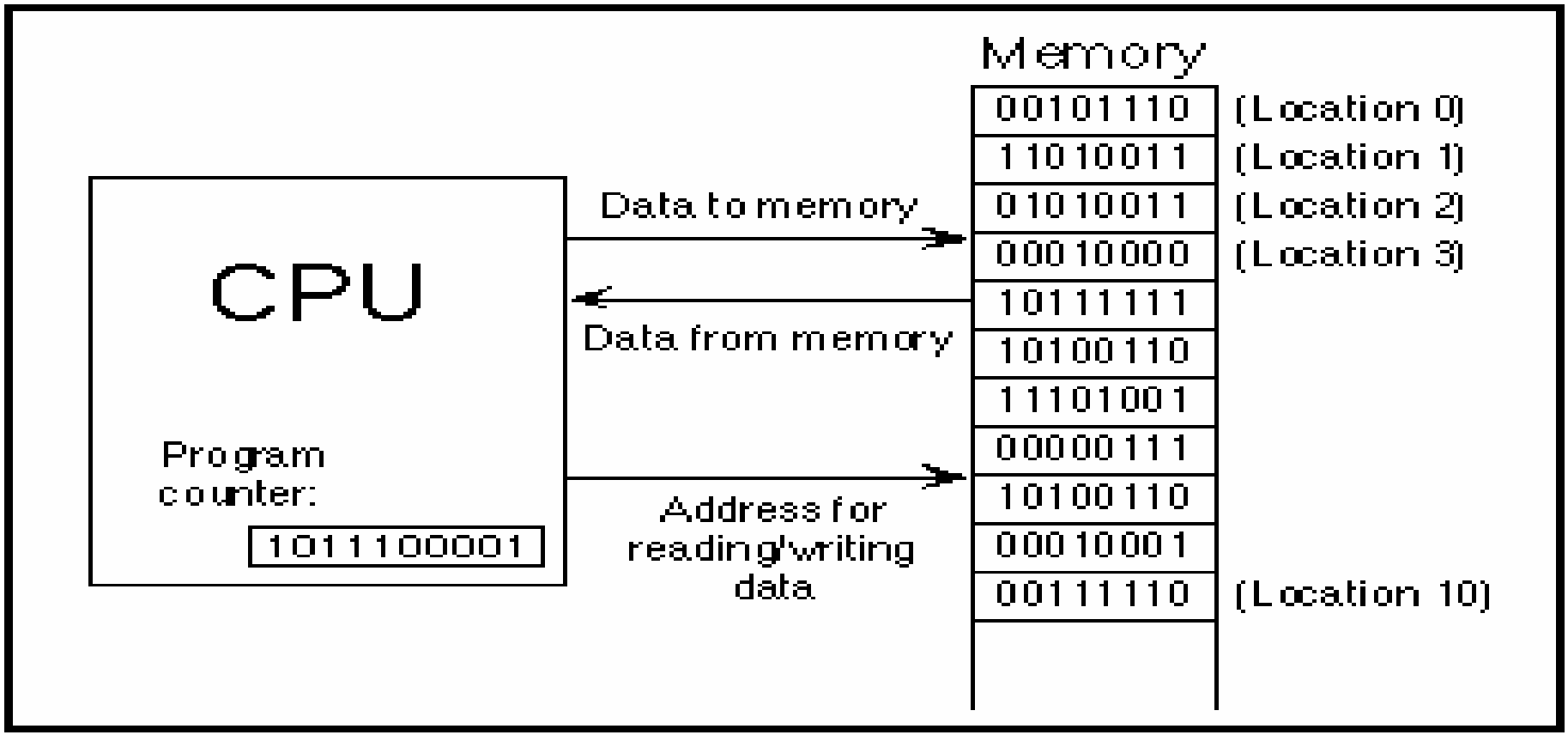
Structure of a Computer

- **Central Processor (CPU)** – principal component of a computer; does all the work
- **Main Memory** – holds data and programs that are in active use
- **External Memory** – for long term storage (hard disk, floppy, etc.)
- **External devices** - monitor, keyboard, etc.

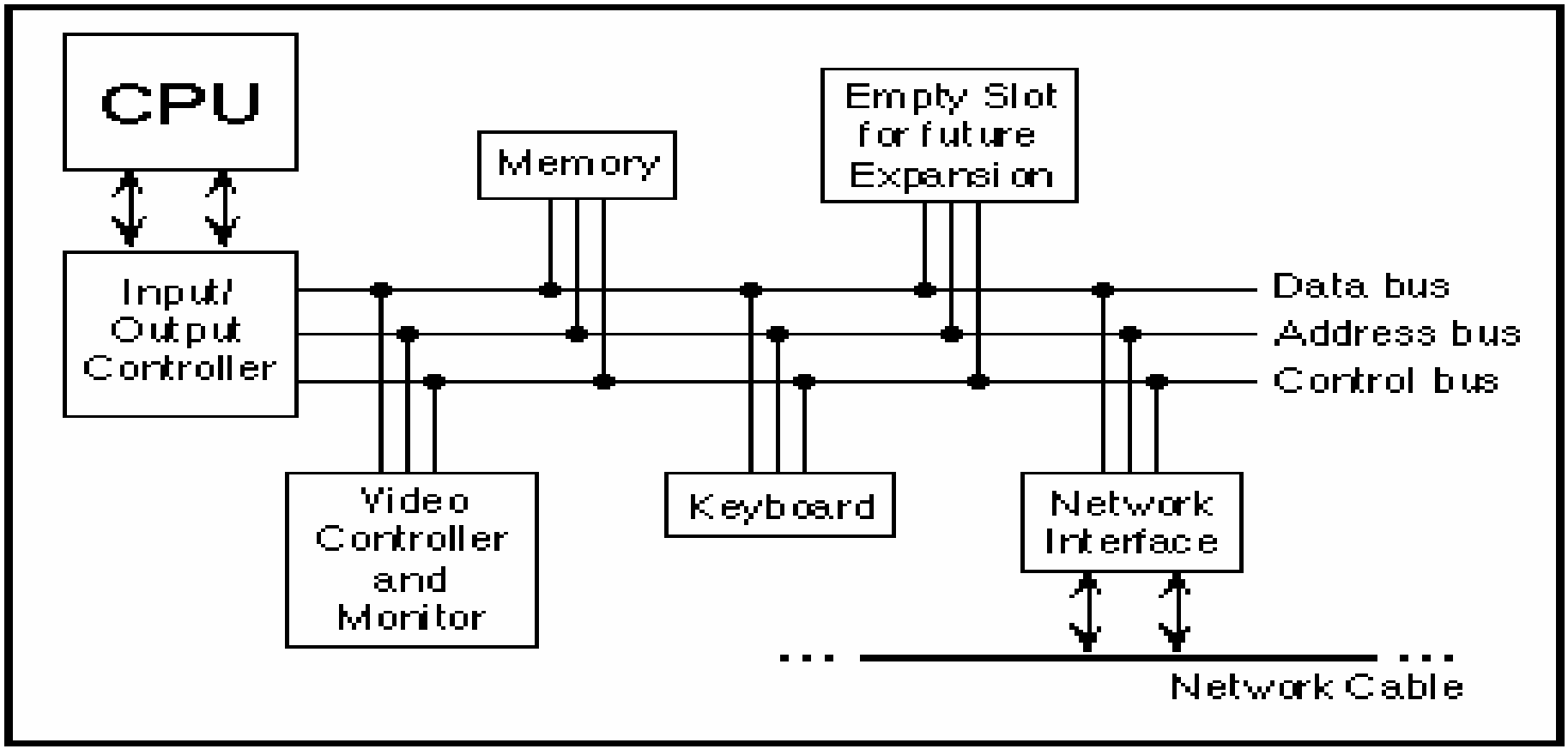




Central Processor (CPU) and memory



CPU and other appliances



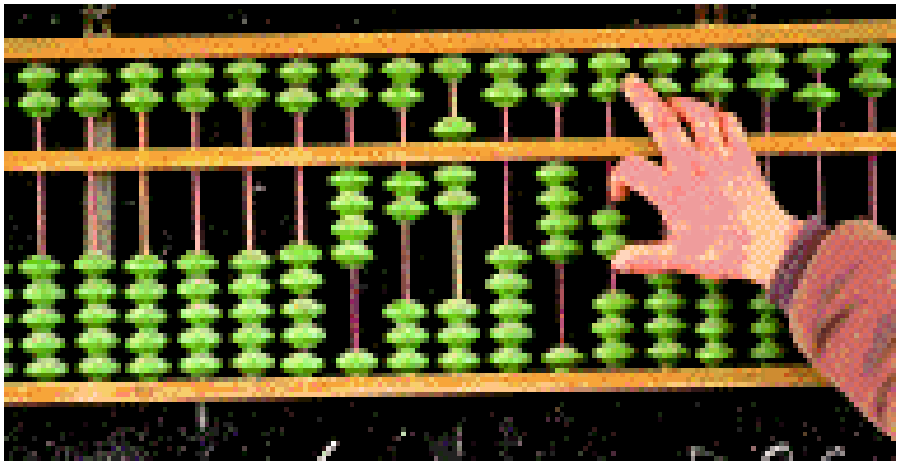
What Is (and Isn't) a Computer

- A **computer** is a device which takes data in one form, uses it, and produces a different form of information which is related to (but not the same as) the original data.

Univ. of Virginia

What Is (and Isn't) a Computer

- The Abacus is not a computer by our definition.
- It is an early calculation device that only holds numbers for the person using it.



Univ. of Virginia

What Is (and Isn't) a Computer

- Stonehenge is a computer by our definition.
- It takes the movement of the planets, sun and other heavenly bodies and provides information concerning eclipses and other astronomical events.

Univ. of Virginia

What Is (and Isn't) a Computer

- The bathroom scale is a computer by our definition.
- It takes in the amount of gravitational pull between a human body and the earth and provides us with the amount of pounds or kilograms.

Univ. of Virginia

What Is (and Isn't) a Computer

- A calculator is a computer by our definition.
- They range from doing simple arithmetic to powerful models that produce graphic output.

The Many Kinds of Computers

- **The three major comparisons of computers are:**
  - Electronic computers versus Mechanical computers
  - General-purpose versus Special-purpose computers
  - Digital versus Analog computers

The Many Kinds of Computers

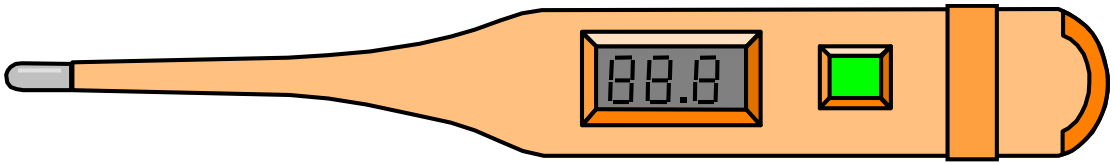
- **Electronic Computers**
  - Constructed from transistors that use electricity to function.
- **Mechanical Computers**
  - Do not use electricity to function.
  - Constructed of a combination of gears, levers and springs.

The Many Kinds of Computers

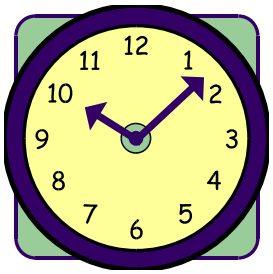
- **General-purpose Computers**
  - Were not manufactured to do any one thing.
  - Changeable to do any task.
- **Special-purpose Computers**
  - Manufactured to do a predetermined task or set of tasks.

The Many Kinds of Computers

- **Digital Computers**
  - One that functions in discretely varying quantities.
  - Produces or gives results that are also discretely varying.



- **Analog Computers**
  - One that functions in continuously varying quantities.
  - Produces or gives results that are also continuously varying.
  - Numerical data are represented by measurable physical variables, such as electrical voltage.



The General-Purpose Digital Computer

- **The General-Purpose Digital Computer**
  - Accepts information of many kinds.
  - Changes it in a way that is controlled by humans.
  - Presents results in a way usable by humans.

What Is Information?

- **The five types of information that are the only types the computer commonly manipulates:**
  - Visual (pictures)
  - Numeric (numbers)
  - Character (text)
  - Audio (sound)
  - Instructions (programs)

What Is Information?

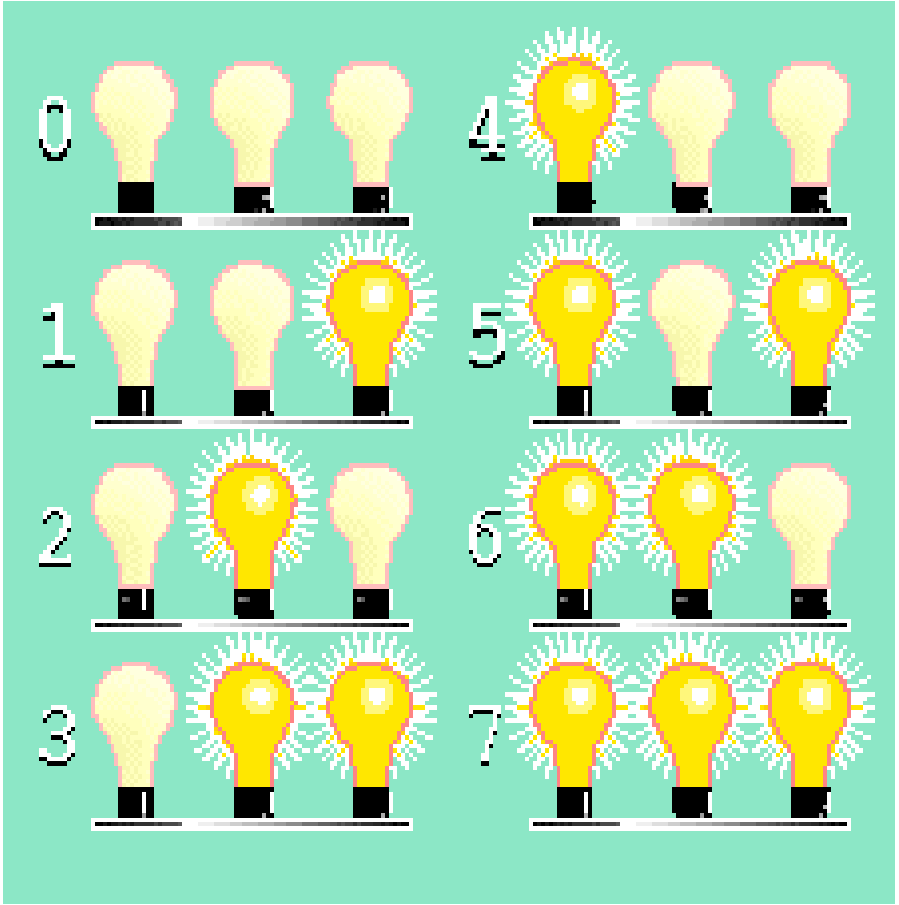
- **Before the computer can use any type of information, it must be stored in the computer's memory.**
- **Problem: How is information stored within the computer?**
  - Information is stored in numerical form within the computer
  - Modern computers work in a system of numbers called binary numbers

What Is Information?

- Binary numbers:
  - Similar to familiar decimal system.
  - Uses only two symbols: 0 and 1.
  - The choice of using binary numbers is dictated by cost and reliability.
- Binary circuits:
  - Electronic circuits are cheapest and most reliable if they only assume two states or conditions.
  - These binary circuits have only two states, ON or OFF.

Representation of Numbers

- Binary numbers use only two symbols: 0 and 1.
- How can more than two possibilities be represented?
- A three light system can have up to eight combinations. Each combination can represent a code.



Representation of Numbers

- **Binary equivalents of the numerals 0 to 7.**

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Binary numerals

Representation of Numbers

- **The** decimal **system:**

		2	1	0	Position
		100s	10s	1s	Weight
		↓	↓	↓	
312	=	3	1	2	$= (3x\ 10^2) + (1x10^1) + (2x10^0)$
					$= (3x100) + (1x10) + (2x1)$
					$= 300 + 10 + 2$

Representation of Numbers

- **The** binary **system:**

$$
\begin{array}{lccc}
 & 4s & 2s & 1s \\
 & \downarrow & \downarrow & \downarrow \\
101_{\text{binary}} = & 1 & 0 & 1
\end{array}
$$

$$= (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$$
$$= (1 \times 4) + (0 \times 2) + (1 \times 1)$$
$$= 4 + 0 + 1$$
$$= 5$$

$101_{\text{binary}} = 5$ in decimal

Representation of Numbers

- **Translate 10011 $_{\text{binary}}$ into a decimal number.**
  http://math.hws.edu/TMCM/java/DataReps/index.html

1. Place the base two numerals under the place values.

2^4	2^3	2^2	2^1	2^0
16	8	4	2	1
1	0	0	1	1

2. Multiply through by the place values.

2^4	2^3	2^2	2^1	2^0
16	8	4	2	1
1	0	0	1	1
1x16	0x8	0x4	1x2	1x1

3. Add up the column for the decimal number.

1
2
0
0
16
19

$10011_{\text{binary}} = 19$ in decimal

Representation of Signed Numbers

- **Signed number representations**
  http://en.wikipedia.org/wiki/Signed_number_representation
- **Sign-and-magnitude**
- **One's complement**
  bitwise NOT
- **Two's complement**
  http://en.wikipedia.org/wiki/2%27s_complement

Two's complement	Decimal
0001	1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4

Two's complement using a 4-bit integer

sign bit									
0	1	1	1	1	1	1	1	=	127
0	0	0	0	0	0	1	0	=	2
0	0	0	0	0	0	0	1	=	1
0	0	0	0	0	0	0	0	=	0
1	1	1	1	1	1	1	1	=	−1
1	1	1	1	1	1	1	0	=	−2
1	0	0	0	0	0	0	1	=	−127
1	0	0	0	0	0	0	0	=	−128

8-bit two's complement integers

Representation of Signed Numbers
Sign-and-Magnitude

- **Sign**
  - representing a number's sign by allocating one sign bit to represent the sign
  - set that bit (often the most significant bit) to *0* for a positive number, and set to *1* for a negative number.
- **Magnitude**
  - The remaining bits in the number indicate the magnitude (or absolute value).
- **Example**
  - byte with only 7 bits (apart from the sign bit) the magnitude can range
    - from 0000000 (0)
    - to 1111111 (127).
  - Thus you can represent numbers from -127_{10} to $+127_{10}$.
  - A consequence of this representation is that there are **two ways** to represent **0**, 00000000 (0) and 10000000 (−0).

Representation of Signed Numbers
1's complement

- bitwise NOT
- the complement of its positive counterpart.
- **two representations of 0**: 00000000 (**+0**) and 11111111 (**−0**).
- To add two numbers represented in this system, one does a conventional binary addition, but it is then necessary to add any resulting carry back into the resulting sum (end-around carry)
- Example:
  −1 (11111110) to +2 (00000010).
  the binary addition alone gives 00000000—not the correct answer!
  Only when the carry is added back in does the correct result (00000001)

Representation of Signed Numbers
2's complement

- Solves problems of
  - multiple representations of 0
  - need for the end-around carry
- Addition of a pair of two's-complement integers is the same as addition of a pair of unsigned numbers => no additional circuitry needed
- Negating a number (whether negative or positive) is done by
  - inverting all the bits and then adding 1 to that result
  - easier method
    - 1. Starting from the right, find the first '1'
    - 2. Invert all of the bits to the left of that one

Two's complement	Decimal
0001	1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4

Two's complement using a 4-bit integer

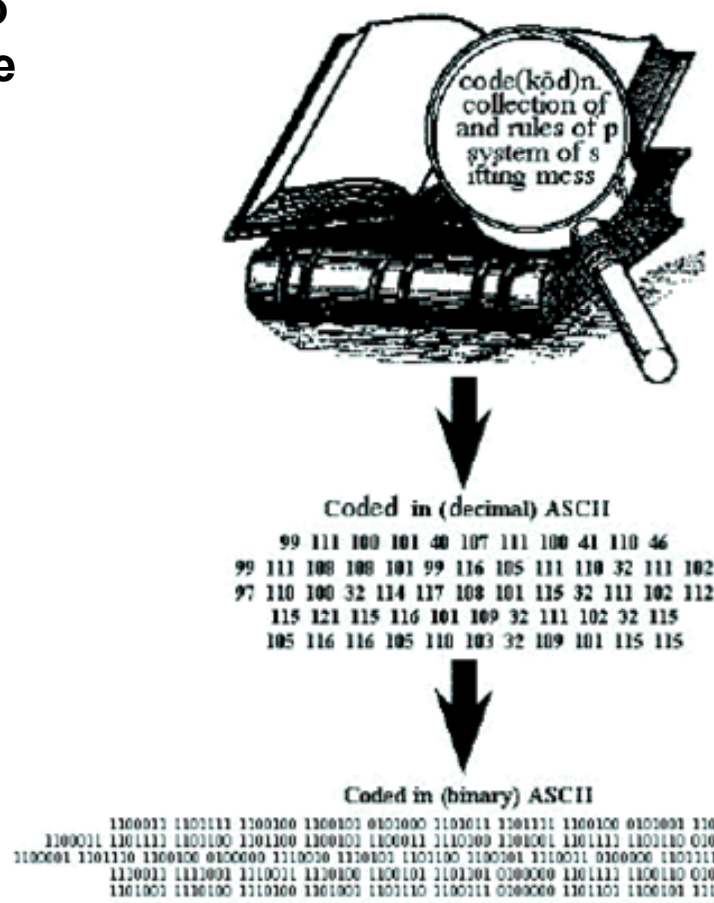
sign bit									
0	1	1	1	1	1	1	1	=	127
0	0	0	0	0	0	1	0	=	2
0	0	0	0	0	0	0	1	=	1
0	0	0	0	0	0	0	0	=	0
1	1	1	1	1	1	1	1	=	−1
1	1	1	1	1	1	1	0	=	−2
1	0	0	0	0	0	0	1	=	−127
1	0	0	0	0	0	0	0	=	−128

8-bit two's complement integers

Representing Symbols and Text

- **Each letter and symbol in a text document must be translated into a binary number for storage in the computer.**

- **Standardized means of storing these codes:**
  - **ASCII** (American Standard Code for Information Interchange)
  - **EBCDIC** (Extended Binary Coded Decimal Interchange Code)
  - **UNICODE (**Extended ASCII)



Representing Pictures

- **Pictures must be translated into a binary format for storage in the computer.**
  - The picture is broken down into small elements.
  - These elements are called Pixels (Picture Elements).

- Digitizer:
  - A device that converts a picture into a binary format for storage in the computer.
  - Examples of digitizers: scanner, digital camera.

Univ. of Virginia

Representing Pictures

- **Digitized picture of a tiger.**

Representing Pictures

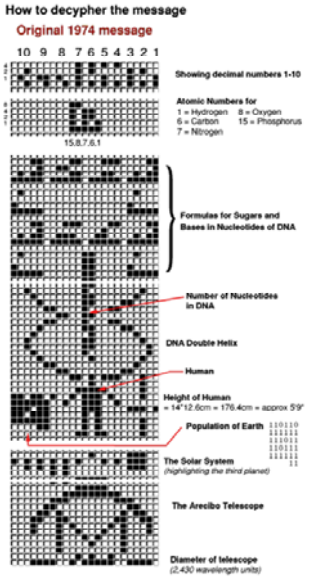
- **Black and white** pixels **are either 0 or 1.**

Representing Pictures

- Gray-Scale:
  - Each pixel contains a value representing some shade of gray.
  - The more shades of gray possible, the more memory will be needed.
    - 4 shades of gray needs 2 bits per pixel:
      - 00, 01, 10, 11
    - 8 shades of gray needs 3 bits per pixel:
      - 000, 001, 010, 011, 100, 101, 110, 111
    - 64 shades of gray needs 6 bits per pixel:
      - 000000, 000001, … 111110, 111111

Univ. of Virginia

Representing Pictures



- **Message transmission from the Arecibo radio telescope in Puerto Rico to other stars on November 16th 1974 and consisted of 1679 pulses of binary code (0's and 1's) which took a little under three minutes to transmit. It was transmitted on a frequency of 2380MHz**
- **broadcast signals possible at a power of up to 20 terawatts (1 terawatt = 1 trillion watts)**
- **This signal was aimed towards the globular star cluster M13, some 25,000 light years away and consisting of some 300,000 stars in the constellation of Hercules.**
- **http://www.cropcircleresearch.com/articles/arecibo.html**
- **http://en.wikipedia.org/wiki/Arecibo_Observatory**

Storage of Binary Information

- **Capacity**

Unit	*Description*	*Approximate Size*
1 bit	1 binary digit	
1 nibble	4 bits	
1 byte	8 bits	1 character
1 kilobyte	1,024 bytes	≈1/2 page, double spaced
1 megabyte	1,048,576 bytes 1 million bytes	≈500,000 pages
1 gigabyte	1,073,741,824 bytes 1 billion bytes	≈5 million pages
1 terrabyte	1 trillion bytes	≈5 billion pages

Instructions as Numbers

- **Fact:**
  - Declarative: statement of being.
  - Imparts knowledge.
- **Instruction:**
  - Imperative: demands action.
  - Controls information or activity.

Instructions as Numbers

- **Instructions:**
  - Must be stored within the computer before use.
  - Must be stored in binary form.
  - A set of binary instructions is called a program.

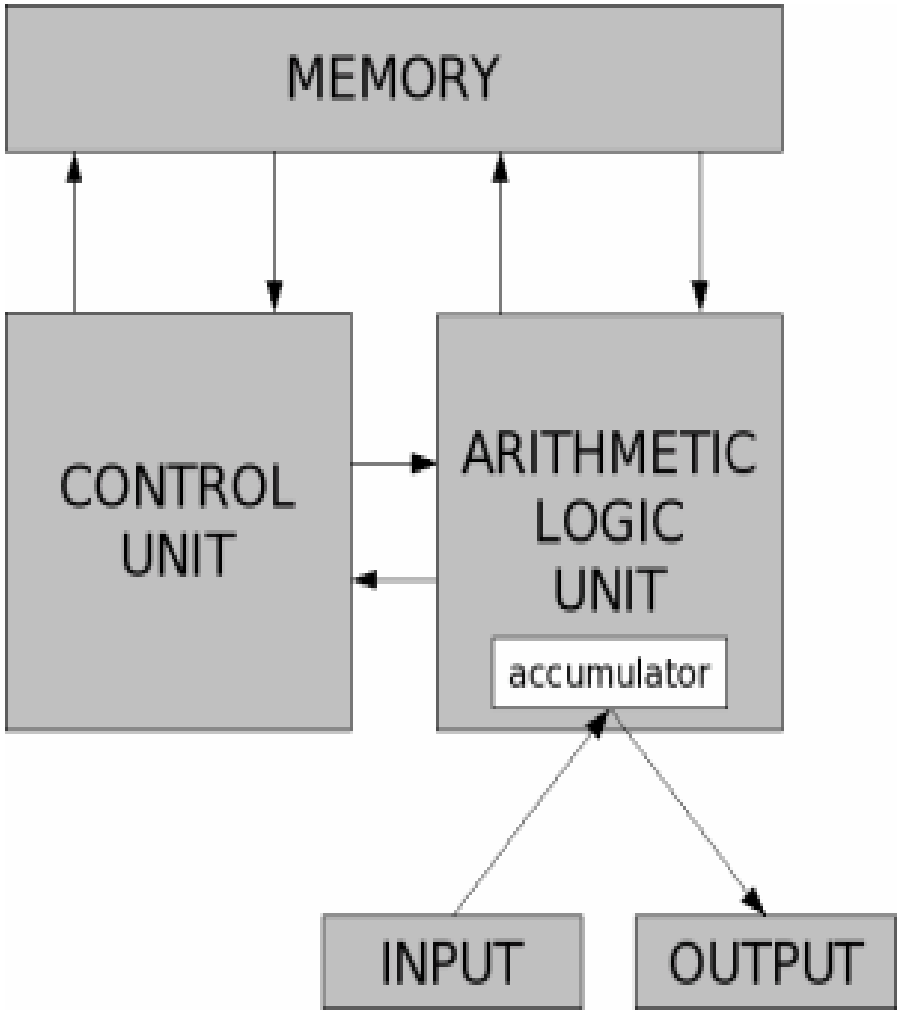
The Stored-program Computer

- **Program:**
  - A collection of instructions for the computer to perform one by one.
- **Machine Language:**
  - The language of the computing machine.
  - All instructions must be in the form of binary numbers (binary code).

The Stored-program Computer

- **Stored-program Computer:**
  - Also known as the von Neumann-type computer.
  - Has memory - a place to keep both:
    - instructions (ie program)
    - and the needed information (ie data)

    needed for computation by the computer.
- **Separation of storage from the processing unit is implicit in the von Neumann architecture.**
- http://en.wikipedia.org/wiki/Von_Neumann_architecture



Conceptual Computer

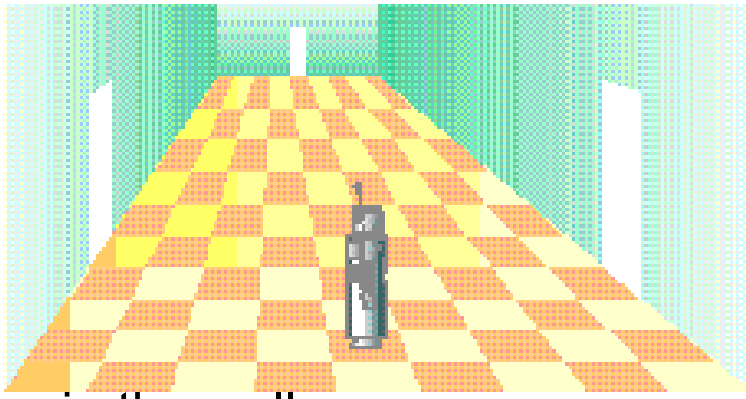
- **In this chapter, we will examine a conceptual computer:**
  - A robot.
- **Why choose imaginary computers to examine how a computer works?**
  - Today's real computers are incredibly complex machines.
    - Huge memories, Large instruction sets, Nearly limitless options and flexibility.
  - Using conceptual computers allows us to strip away the complexity, revealing the underlying simplicity of operation.

Conceptual Computer

- **How is each conceptual computer similar?**
  - Each is a **stored-program computer** (von Neumann computer)
    - Each contains a minimal configuration of:
      - Input units
      - Memory
      - Central processing unit
      - Output units
  - Each stores a program and the data it needs in its own memory.
  - Each executes instructions sequentially.
  - Each is designed with very limited capabilities. (small memory and instruction set.)

Programs and Algorithms

- **Our first example of the computer: The ROBOT computer**



- **The ROBOT's domain**
  - The room is empty.
  - The room is rectangular.
  - There may be one or more open doorways in the walls.
  - The floor is paved with square tiles with lines between them. The lines are easy to see.
  - The size of the room is unknown to us at any given time.
  - The size of the room does not change during the execution of a program.
  - Doorways will never be located in corners.

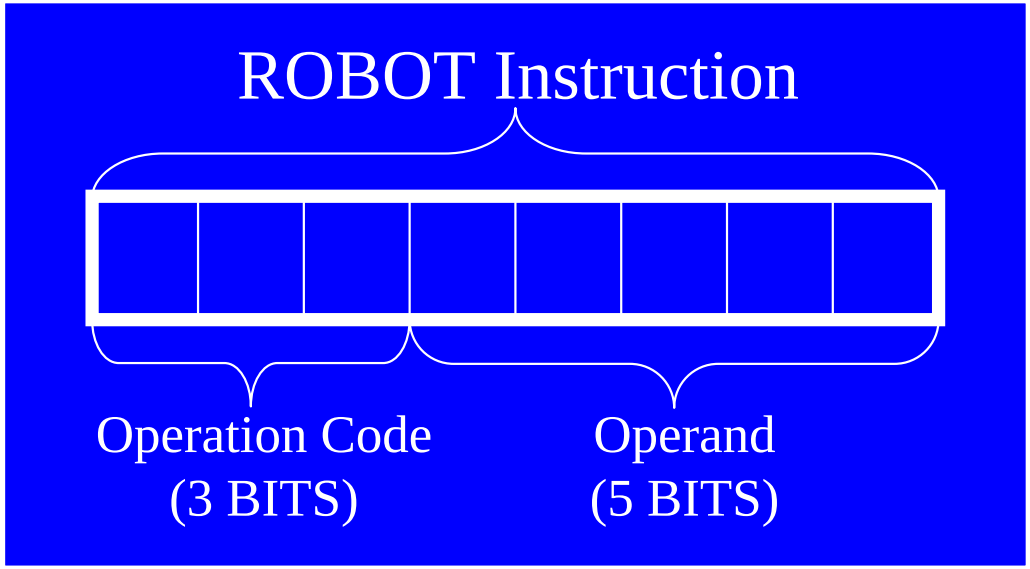
Programs and Algorithms

- **ROBOT Characteristics:**

Forward movement	Moves forward from square to square within its domain.
Changing direction	Turns only 90 degrees to the right.
Arm movement	Can ***raise*** and ***lower*** its arms.
Arm extension	When its arms are raised, they reach to the far edge of the next square.
Sensors	The sensors at the ends of its arms are used to locate walls.
Intelligence	NONE. The ROBOT cannot see or think on its own. It only executed instructions stored in memory.

Programs and Algorithms

- **ROBOT instructions have two parts:**
  - **Operation Code** (Opcode) - Dictates the action to be performed by the ROBOT.
  - **Operand** (Argument) - The address of a position in memory.
  - Each part of a ROBOT instruction is called a **field**.



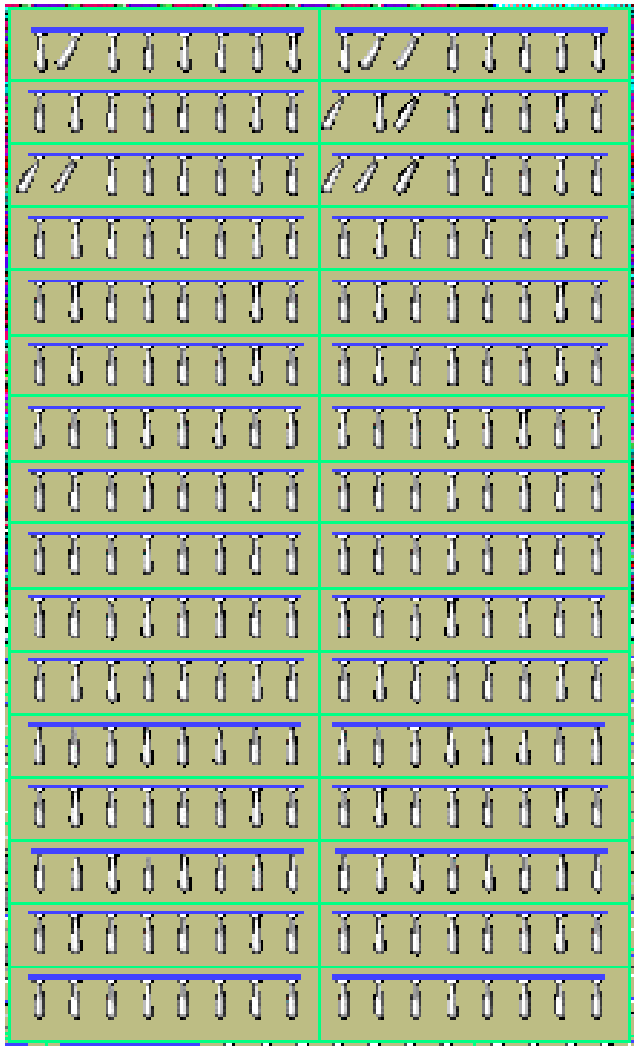
Programs and Algorithms

- **ROBOT Programs:**
  - Lists of instructions can be determined and changed by the person who operates the ROBOT.
  - **Program**: Refers to the list of instructions given to the ROBOT.
    - A program must be placed into the ROBOT's memory before any execution can take place.

- **ROBOT's Memory:**
  - Located on the ROBOT's torso.
  - 32 memory locations.
  - Each memory location is a set of 8 toggle switches.
  - On = 1 Off = 0
  - **Loading** a program: setting the switches.

Programs and Algorithms

- The ROBOT's memory contains 32 memory locations numbered 0 to 31.
- Each location is capable of storing one ROBOT instruction.



Programs and Algorithms

Opcode	English	Action taken by command
000	STEP	The ROBOT takes one STEP forward if possible.
001	TURN	The ROBOT pivots 90 degrees to the right.
010	RAISE	The ROBOT raises its arms if possible. IF NOT POSSIBLE: There MUST be a wall directly in front of the ROBOT. The warning light will come on. *No other commands will be recognized until the light is turned off.*
011	LOWER	The ROBOT lowers its arms if they are raised.
100	SENSE	The ROBOT, with its arms in ***raised*** position, can detect if it is one step away from the wall it is facing. IF IT IS, the warning light will turn on. *Recognizes no other commands until the light is turned off.*
101	GOTO	The ROBOT takes the next command out of normal order. The **Operand**, the last 5 bits of the instruction, tells which memory location is to be performed next.
110	LIGHT	*IF the light is turned on, this command turns it off.* The ROBOT will again recognize instructions in the program.
111	STOP	The ROBOT shuts off its own power.

Programs and Algorithms

- Algorithm:
  - A step-by-step process used to solve a problem.
  - The general solution to the problem.
  - Usually implemented by a program.

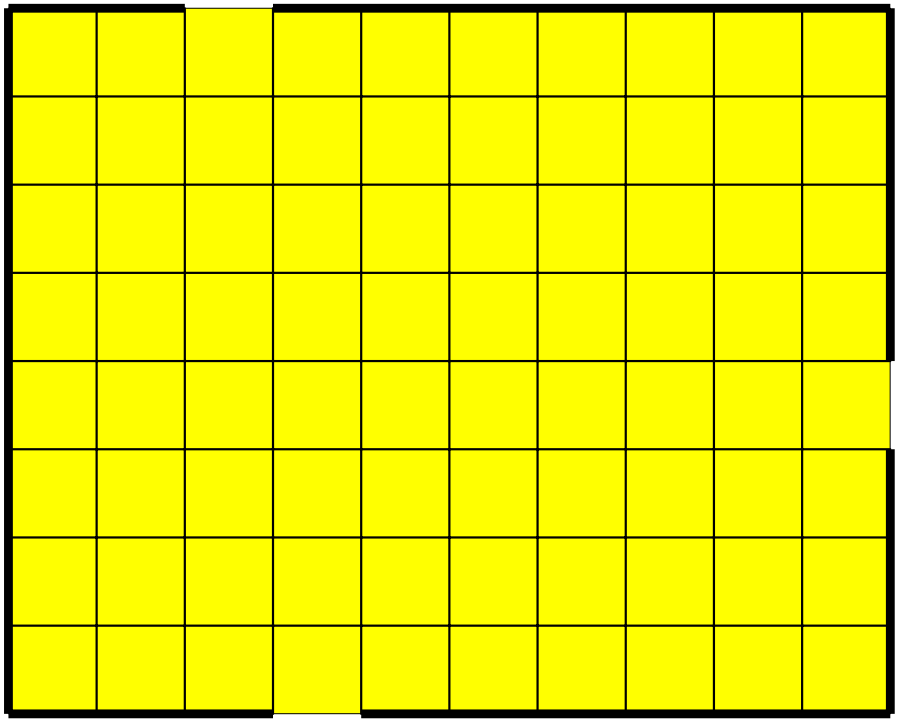
  - Problem: Cause the ROBOT to walk to the wall it is initially facing and then stop with its arms lowered and facing against the wall. Assume the ROBOT is not initially facing an open doorway.

    - Remember:
      - We have NO IDEA how big the room is!
      - We CAN'T just tell it to STEP X-number of times!
      - The algorithm has a general solution. (Solves the problem in all situations.)

Programs and Algorithms

- **Why isn't this a "good enough" solution to the problem of finding the wall in front of the ROBOT?**

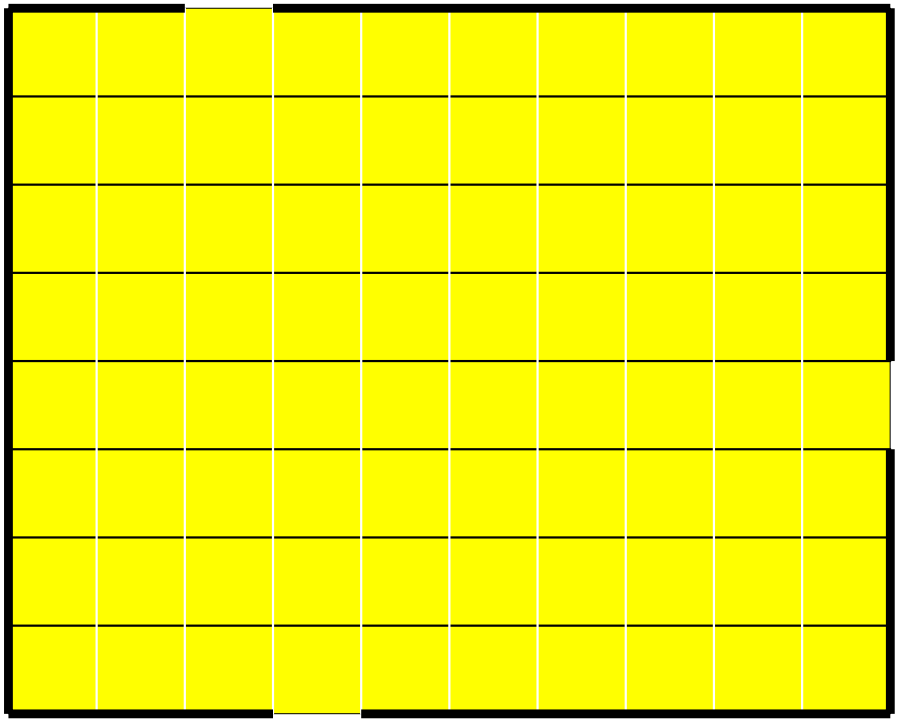
  - 0 RAISE
  - 1 SENSE
  - 2 STEP
  - 3 GOTO 1
  - 4 LIGHT
  - 5 STOP



Programs and Algorithms

- **Why is this a better solution to the problem of finding the wall in front of the ROBOT?**

  - 0 RAISE
  - 1 LOWER
  - 2 STEP
  - 3 GOTO 0
  - 4 LIGHT
  - 5 STOP



Programs and Algorithms

- **Programming the ROBOT - Taking the "English" steps and writing them in the language the ROBOT understands (Machine Language).**
  - **Machine Language** - Written in binary code, the program is in the form the computer understands.

"English" Version	Machine Language Version
RAISE	01000000
LOWER	01100000
STEP	00000000
GOTO 0	10100000
LIGHT	11000000
STOP	11100000

Programs and Algorithms

- **Loop** - A sequence of instructions which is repeated one or more times when a program is executed.

- **Infinite loop -** A set of instructions which causes the program to repeat the same commands over and over with no possible way of stopping.

Programs and Algorithms

- **Cause the ROBOT to walk around the perimeter of the room.**
  - Does the program ever stop? What kind of loop does this program contain?

  - 0 RAISE
  - 1 LOWER
  - 2 STEP
  - 3 GOTO 0
  - 4 LIGHT
  - 5 TURN
  - 6 GOTO 0
  - 7 STOP

How does a person communicate with a computer?

- Programming languages bridge the gap between human thought process and computer binary circuitry.